Updated S 09/360,542

L Number	Hits	Search Text	DB	Time stamp
Number 1	15753	application adj program\$4 adj interface	USPAT;	2004/03/11
•		approduction and programmy can, income	US-PGPUB;	11:44
			EPO; JPO;	
			DERWENT;	
			IBM_TDB	
2	30774	automatic\$4 with (backup or back-up or	USPAT;	2004/03/11
		recover\$6 or restor\$6 or (back adj1 up))	US-PGPUB;	12:37
			EPO; JPO;	
			DERWENT;	İ
			IBM_TDB	
3	23053	execut\$6 near2 state\$	USPAT;	2004/03/11
			US-PGPUB;	11:45
			EPO; JPO;	
			DERWENT;	
			IBM_TDB	
4	0	(application adj program\$4 adj interface)	USPAT;	2004/03/11
		same (automatic\$4 with (backup or back-up	US-PGPUB;	11:28
		or recover\$6 or restor\$6 or (back adj1 up)))	EPO; JPO;	
	;	same (execut\$6 near2 state\$)	DERWENT;	
			IBM_TDB	
5	97	(application adj program\$4 adj interface)	USPAT;	2004/03/11
		and (automatic\$4 with (backup or back-up	US-PGPUB;	11:28
		or recover\$6 or restor\$6 or (back adj1 up)))	EPO; JPO;	
		and (execut\$6 near2 state\$)	DERWENT;	
			IBM_TDB	
6	10	(application adj program\$4 adj interface)	USPAT;	2004/03/11
		same (automatic\$4 with (backup or back-up	US-PGPUB;	11:29
		or recover\$6 or restor\$6 or (back adj1 up)))	EPO; JPO;	
	:		DERWENT;	
_			IBM_TDB	0004/00/44
7	0	((application adj program\$4 adj interface)	USPAT;	2004/03/11
		same (automatic\$4 with (backup or back-up	US-PGPUB;	11:29
		or recover\$6 or restor\$6 or (back adj1 up))))	EPO; JPO;	
		and (execut\$6 near2 state\$)	DERWENT;	
8	39	(application adj program\$4 adj interface)	IBM_TDB USPAT;	2004/03/11
0	39	same (execut\$6 near2 state\$)	US-PGPUB;	11:29
		same (executo near2 states)	EPO; JPO;	11:29
			DERWENT;	
			IBM_TDB	
9	0	((application adj program\$4 adj interface)	USPAT;	2004/03/11
-		same (execut\$6 near2 state\$)) and	US-PGPUB;	11:29
		(automatic\$4 with (backup or back-up or	EPO; JPO;	
		recover\$6 or restor\$6 or (back adj1 up)))	DERWENT;	
			IBM_TDB	
10	14	((application adj program\$4 adj interface)	USPAT;	2004/03/11
-		same (execut\$6 near2 state\$)) and (backup	US-PGPUB;	11:31
	! 	or back-up or recover\$6 or restor\$6 or (back	EPO; JPO;	
		adj1 up))	DERWENT;	
			IBM_TDB	

11	34069	(application adj program\$4 adj interface) or	USPAT;	2004/03/11
		API	US-PGPUB;	11:44
			EPO; JPO;	
			DERWENT;	
			IBM_TDB	
12	13338	execut\$6 near1 state\$	USPAT;	2004/03/11
		,	US-PGPUB;	11:45
			EPO; JPO;	
			DERWENT;	
			IBM_TDB	
13	14992	(backup or back-up or recover\$6 or restor\$6	USPAT;	2004/03/11
	14332	or (back adj1 up)) near2 (program\$ or	US-PGPUB;	11:47
		software or application)	EPO; JPO;	11.77
		Software or application)	1	
			DERWENT;	
			IBM_TDB	0004/00/44
14	70	((application adj program\$4 adj interface) or	USPAT;	2004/03/11
		API) and (execut\$6 near1 state\$) and	US-PGPUB;	11:48
		((backup or back-up or recover\$6 or	EPO; JPO;	
		restor\$6 or (back adj1 up)) near2 (program\$	DERWENT;	
		or software or application))	IBM_TDB	
15	0	((application adj program\$4 adj interface) or	USPAT;	2004/03/11
		API) same (execut\$6 near1 state\$) same	US-PGPUB;	11:48
		((backup or back-up or recover\$6 or	EPO; JPO;	
		restor\$6 or (back adj1 up)) near2 (program\$	DERWENT;	
		or software or application))	IBM_TDB	
16	17008	((application adj program\$4 adj interface) or	USPAT;	2004/03/11
		API) same e ((backup or back-up or	US-PGPUB;	11:49
		recover\$6 or restor\$6 or (back adj1 up))	EPO; JPO;	
		near2 (program\$ or software or application))	DERWENT;	
			IBM_TDB	
17	123	((application adj program\$4 adj interface) or	USPAT;	2004/03/11
•		API) same ((backup or back-up or recover\$6	US-PGPUB;	11:49
		or restor\$6 or (back adj1 up)) near2	EPO; JPO;	
		(program\$ or software or application))	DERWENT;	
		(programe or software or approacion))	IBM_TDB	
18	1	(((application adj program\$4 adj interface)	USPAT;	2004/03/11
	1	or API) same ((backup or back-up or	•	11:49
			US-PGPUB;	11.73
		recover\$6 or restor\$6 or (back adj1 up))	EPO; JPO;	
		near2 (program\$ or software or	DERWENT;	
		application))) and (execut\$6 near1 state\$)	IBM_TDB	
19	3569	automatic\$4 adj2 recover\$4	USPAT;	2004/03/11
			US-PGPUB;	11:51
			EPO; JPO;	
			DERWENT;	
			IBM_TDB	
20	124	(automatic\$4 adj2 recover\$4) same restor\$6	USPAT;	2004/03/11
			US-PGPUB;	11:52
			EPO; JPO;	
			DERWENT;	
	1		IBM_TDB	

04	T	//	HCDAT.	2004/02/44
21	1	((automatic\$4 adj2 recover\$4) same restor\$6) and ((application adj program\$4 adj interface) or API)	USPAT; US-PGPUB; EPO; JPO; DERWENT;	2004/03/11
22	136	(automatic\$4 adj2 recover\$4) and	IBM_TDB USPAT;	2004/03/11
		(application adj program\$4 adj interface)	US-PGPUB; EPO; JPO; DERWENT;	12:14
			IBM_TDB	
23	0	((automatic\$4 adj2 recover\$4) same restor\$6) and (application adj program\$4 adj interface)	USPAT; US-PGPUB; EPO; JPO; DERWENT;	2004/03/11 12:15
			IBM_TDB	
24	78	(automatic\$4 adj2 recover\$4) and restor\$6 and (application adj program\$4 adj interface)	USPAT; US-PGPUB; EPO; JPO; DERWENT;	2004/03/11 12:19
25	18	((automatic\$4 adj2 recover\$4) and restor\$6 and (application adj program\$4 adj	IBM_TDB USPAT; US-PGPUB;	2004/03/11 12:16
		interface)) and (714/\$).ccls.	EPO; JPO; DERWENT; IBM_TDB	
26	32	(automatic\$4 adj2 recover\$4) and (restor\$6 adj2 (program or software or application))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/03/11 12:40
27	4	((automatic\$4 adj2 recover\$4) and (restor\$6 adj2 (program or software or application))) and (application adj program\$4 adj interface)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/03/11 12:21
28	3734	(backup or back-up or (back\$4 adj1 up)) near2 (software\$ or program\$ or application\$)	USPAT; US-PGPUB; EPO; JPO; DERWENT;	2004/03/11 12:57
29	196	((backup or back-up or (back\$4 adj1 up)) near2 (software\$ or program\$ or	IBM_TDB USPAT; US-PGPUB;	2004/03/11 12:40
		application\$)) and (restor\$6 adj2 (program or software or application))	EPO; JPO; DERWENT; IBM_TDB	
31	9526	(hardware or execut\$6) adj1 state\$	USPAT; US-PGPUB; EPO; JPO; DERWENT;	2004/03/11 12:41

32	1	((((backup or back-up or (back\$4 adj1 up))	USPAT;	2004/03/11
32	•	near2 (software\$ or program\$ or	US-PGPUB;	12:42
		application\$)) and (restor\$6 adj2 (program	EPO; JPO;	
		or software or application))) and (application	DERWENT;	
		adj program\$4 adj interface)) and	IBM_TDB	
		((hardware or execut\$6) adj1 state\$)		
30	16	(((backup or back-up or (back\$4 adj1 up))	USPAT;	2004/03/11
		near2 (software\$ or program\$ or	US-PGPUB;	12:56
		application\$)) and (restor\$6 adj2 (program	EPO; JPO;	
		or software or application))) and (application	DERWENT;	
		adj program\$4 adj interface)	IBM_TDB	
33	1		USPAT	2004/03/11
				12:54
34	1		USPAT	2004/03/11
				12:54
35	1		USPAT	2004/03/11
				12:55
36	1		USPAT	2004/03/11
				12:55
37	1		USPAT	2004/03/11
				12:55
38	1		USPAT	2004/03/11
				12:55
39	20539	restor\$6.ti.	USPAT;	2004/03/11
			US-PGPUB;	12:58
			EPO; JPO;	
			DERWENT;	
			IBM_TDB	
40	113580	recover\$6.ti.	USPAT;	2004/03/11
	·		US-PGPUB;	12:58
			EPO; JPO;	
			DERWENT;	
44	47740		IBM_TDB	2004/02/44
41	17718	(backup or back-up or (back\$4 adj1 up)).ti.	USPAT;	2004/03/11
			US-PGPUB;	12:58
			EPO; JPO; DERWENT;	
			IBM_TDB	
42	0	restor\$6.ti. and recover\$6.ti. and ((backup	USPAT;	2004/03/11
		or back-up or (back\$4 adj1 up)).ti.) and	US-PGPUB;	12:58
		(application adj program\$4 adj interface)	EPO; JPO;	
		(abbasement and bredigment and morning)	DERWENT;	
			IBM TDB	
43	112780	restor\$6.ab.	USPAT;	2004/03/11
			US-PGPUB;	12:58
			EPO; JPO;	
			DERWENT;	
			IBM_TDB	
44	327436	recover\$6.ab.	USPAT;	2004/03/11
			US-PGPUB;	12:58
			EPO; JPO;	
			DERWENT;	
			IBM_TDB	

45	49561	(backup or back-up or (back\$4 adj1 up)).ab.	USPAT;	2004/03/11
70	1300.	(Sackap of Sack ap of (Sackap au), ap), ass	US-PGPUB;	12:59
			EPO; JPO;	
			DERWENT;	
			IBM_TDB	
46	53855	(application adj program\$4 adj interface)	USPAT:	2004/03/11
		and restor\$6.ab. and recover\$6.ab. adn	US-PGPUB;	12:59
		((backup or back-up or (back\$4 adj1	EPO; JPO;	
		up)).ab.)	DERWENT;	
		1,,,,	IBM_TDB	
47	4	(application adj program\$4 adj interface)	USPAT;	2004/03/11
		and restor\$6.ab. and recover\$6.ab. and	US-PGPUB;	13:01
		((backup or back-up or (back\$4 adj1	EPO; JPO;	
		up)).ab.)	DERWENT;	
			IBM_TDB	
48	2	(registry or catalog) and ((application adj	USPAT;	2004/03/11
		program\$4 adj interface) and restor\$6.ab.	US-PGPUB;	13:03
		and recover\$6.ab. and ((backup or back-up	EPO; JPO;	
		or (back\$4 adj1 up)).ab.))	DERWENT;	
			IBM_TDB	
49	30	(registry or catalog) and recover\$6.ab. and	USPAT;	2004/03/11
		((backup or back-up or (back\$4 adj1	US-PGPUB;	13:03
		up)).ab.)	EPO; JPO;	
			DERWENT;	
			IBM_TDB	
50	4	((registry or catalog) and recover\$6.ab. and	USPAT;	2004/03/11
		((backup or back-up or (back\$4 adj1	US-PGPUB;	13:04
		up)).ab.)) and ((application adj program\$4	EPO; JPO;	
		adj interface) or API)	DERWENT;	
			IBM_TDB	

US-PAT-NO:

6564215

DOCUMENT-IDENTIFIER:

US 6564215 B1

TITLE:

Update support in database content management

----- KWIC -----

Abstract Text - ABTX (1):

A computer system updates a data object that is maintained in data storage

external to a database management system (DBMS), after receiving an update

request from a DBMS client for the data object, by first scheduling the update

request with the DBMS to provide access to the external data object, then $\ensuremath{\mathsf{DBMS}}$

initiating a subtransaction in the DBMS for the update request, updating the

data object with an in-place update action at the external data storage to

thereby produce an updated data object and also updating the DBMS $\!$ metadata of

the data object, next appending information relating to type and time of the $% \left(1\right) =\left(1\right) \left(1\right) +\left(1\right) \left(1\right) \left(1\right) +\left(1\right) \left(

update action in an object version table, and then executing a $\underline{\textbf{backup}}$ operation

of the updated data object. This permits update-in-place operations on the $% \left(1\right) =\left(1\right) +\left(1\right)$

external data object, under supervision of the DBMS. The system thereby

supports update-in-place operations on external data with access control,

backup and recovery,
database and transaction consistency in accordance with a

management system, while avoiding large copy operations that would consume

network resources.

Detailed Description Text - DETX (2):

FIG. 1 is a representation of the functional components contained in a

computer system 100 constructed in accordance with the present invention. The $\,$

computer system includes the datalinks engine 102 of a database management

system (DBMS) 104 at a first node of a computer network, and two file sites

106, 108 communicating with the DBMS over the network 110. Data objects 109 $\,$

external to the DBMS 104. A DBMS client 112 at another network node

113 also communicates with the DBMS 104 over the network. In accordance with invention, a computer program application 114 at the client node 113 updates a data object 109 stored at one of the file sites 106, 108 by directly accessing and updating the data object in the external store through a file system application program interface (API) 116 at each respective file site without first making a copy of the data object. The client application 114 also accesses metadata of the data object through an SQL interface provided by the DBMS 104 to maintain consistency between the data object and its metadata. This technique permits update-in-place operations on the data object, under supervision of the DBMS 104. In this way, the computer system 100 supports update operations on external data with access control, backup and recovery, and transaction consistency in accordance with the DBMS, while avoiding large copy operations that would consume network resources and also maintaining the

Detailed Description Text - DETX (5):

same object name from the file system perspective.

In a conventional DBMS system, the file will be checked out via the DBMS $\,$

104, and a copy of a checked out data object 109 will be sent from the file

site 106 to the client 112, where the application 114 can be used for update

operations on the copy. Following the update operations, the client 112

returns (checks in) the updated data object copy to the file site 106, whereupon the updated data object 109 is stored. As noted above, however, the

computer system 100 of the present invention can support update-in-place

operations by providing a system having a DBMS in which client applications 114

can directly update data objects through the $\underline{\mathtt{API}}$ 116 of the file system at an

external file site 106. The applications 114 can be any one of a variety of

client applications, including applications that provide word processing,

spreadsheet, database, and Internet-protocol web browser functions. The direct

updating from the client application to the external file system API is represented in FIG. 1 by the dashed line 121.

Detailed Description Text - DETX (24):

After the DLFM processing of the Update_Pending request, the DLFS continues

the update-in-place processing by passing the file open request to the native

file system at the external store. The requested file is then opened by the

native file system and a file pointer is returned to the client. This operation is represented by the FIG. 4 flow diagram box numbered 410. All

subsequent update processing on the requested file, including write and read

operations, will then be carried out by the client application without any

further intervention by either the DLFM or DLFS. Such subsequent update

processing can be carried out using the $\underline{\mathtt{API}}$ of the external store file system.

These update operations are represented in FIG. 4 by the flow diagram $\ensuremath{\mathsf{box}}$

numbered 412.

Detailed Description Text - DETX (33):

When a "linked" file (a file whose access is under database system control)

is updated, the preferred embodiment makes an archive copy of the updated file.

Each archived version of the updated file is associated with a unique identifier and the information is stored in the File_Version table. In a

restore operation, after restoring the database to a condition at a specific

point in time in the past, the corresponding files would also have to be

restored from the archive to match the restored database state, or condition.

To do so, the database system can start a reconciliation process to synchronize

the database state and the file system state. The following are the steps the

system performs for such reconciliation processing. (a) A database agent sets

an identifier (called db_state_id) for the current database state which can be

implemented by a time stamp or tail LSN. (b) The database agent scans the

 $\frac{\mathtt{catalog}}{\mathtt{with}\ \mathtt{a}}$ tables to identify all user tables having one or more columns

Datalink data type that support the update-in-place processing described

herein. (c) For each table found, the database agent scans the table and

extracts the datalink column value (URL) from each record. (d) The agent

inserts the extracted URL into a message block and sends it, along with the

database state identifier db_state_id, to a DLFM daemon process. (e)

The DLFM
daemon process uses the URL and db_state_id to look up the
corresponding entry
in a DLFM table (the File_Version table) and then extracts the archive
file
name from the DLFM table entry. When there is more than one entry with
the
same URL, the one with the greatest value of unique recovery id that is
smaller
than or equal to the received db_state_id is selected. (f) The DLFM
daemon
process then restores the file to the file system from the archive
device if
the file is not already in the file system.

Detailed Description Text - DETX (35):

FIG. 7 is a flow diagram that illustrates the processing steps executed by

the computer processing system 100 of FIG. 1 to implement the coordinated

database restore operation. The flow diagram illustrates the operation steps

described above. Thus, in the first processing step, a database agent sets an

identifier for the current database state, which can include time stamp data.

This processing is represented by the flow diagram box numbered 702, and

corresponds to step (a) above. Next, the database agent scans the catalog

tables to identify all user tables having one or more columns with Datalink

data types that support the update-in-place processing described herein. This

step is represented by box 704 in FIG. 7, and step (b) above. Next, for each

table found, the database agent scans the table and extracts the URL from each

record, as represented by box 706 and step (c). In the processing represented

by box 708 of FIG. 7 (and step (d) above), the database agent inserts the

extracted URL into a message block and sends it, along with the database state

identifier db_state_id, to a DLFM daemon process.

US-PAT-NO:

6014437

DOCUMENT-IDENTIFIER:

US 6014437 A

TITLE:

Multi service platform architecture for

telephone

networks

----- KWIC -----

Detailed Description Text - DETX (41):

The call processors (CPs)--consisting of a pair of RISC System/6000 computers based on the IBM RISC System/6000 workstation--execute layered

software (see discussion of software architecture in next section below) that

includes the logic determining how calls are treated and processed. Both $\ensuremath{\mathtt{CPs}}$

are connected to and share the same physical disks for data storage, and ${\rm data}$

stored on the disks is available to both. The CPs are configured in a high

availability arrangement wherein both are active and processing calls, but

which can fall back to continued operation with a single CP when necessary.

The CPs use "mirrored" disks to ensure reliability and provide quick access to

data. Each processor has a CD-ROM drive and tape drive for loading its respective operating system $\frac{\text{software and other software and for backing up}}{\text{up}}$ and

restoring information accessed principally through the mirrored disks.

Detailed Description Text - DETX (47):

FIG. 3 provides an overview of the software architecture presently contemplated for the MSP/6000 system. The software is architected to operate $\frac{1}{2}$

in three layers--a service application management layer 30, a service application layer 31, and a communications and management layer 32--based on a

platform 33 whose major component is the AIX/6000 Operating System commonly

used in RISC/6000 processors. An <u>application programming interface</u> associated

with service creation (Service Creation API) 35 links layers 31 and 32.

Detailed Description Text - DETX (48):

The Application Management layer 30--which is intended to be shared by

multiple MSP/6000 nodes, and may be located physically and geographically $% \left(1\right) =\left(1\right) \left(1\right)$

separate from any and all of the nodes--consists of an element

management system (EMS) 40 and a service creation environment (SCE) 41. SCE 41 provides both a graphical user interface (GUI) and a text-based application oriented language (AOL) suited for the development, debugging and testing of telephony service application logic (e.g. a language based on the DirectTalk/6000 state table service language). Using such language, developers can create applications that play back selected pre-recorded announcements serving as prompts to a user; such prompts being either prerecorded and edited from a tape cassette unit or recorded live using a microphone. EMS 40 includes tools for distributing, backing up and restoring service applications (including recorded prompts). The recorded prompts are distributed/restored to call processors in (multiple) MSP/6000 systems and transferred by the latter processors to voice peripherals in respective systems. Tools for backing up and restoring the prompts are distributed to MOC processors in respective systems and accessed therefrom when needed.